See also:
Proceedings of the First Workshop on Human Computer Interaction for Mobile Devices,

The Impact of Time and Place on the Operation of Mobile Computing Devices

Chris Johnson

Glasgow Accident Analysis Group , Department of Computing Science, University of Glasgow,
Email: johnson@dcs.glasgow.ac.uk

Abstract

Recent improvements in the quality and reliability of wireless communications has led to the development of a range of mobile computing devices. Many portable computers now offer modem connections through cellular and satellite telephone networks. Taxi services, emergency vehicles, domestic repair teams all now rely upon mobile links to central computing systems. In spite of these advances, a number of technical problems still affect the quality of interaction with mobile applications. Electromagnetic interference blocks radio signals. Obstacles in the line of sight can interrupt microwave and infra-red transmissions. Tracking problems frustrate the use of low-level satellites. Transmission delays affect the service provided by higher, geostationary satellites. From the users' point of view, these problems manifest themselves as geographical constraints upon the usability of their 'mobile' device. This lead to delays in the transmission of critical information. These, in turn, lead to the frustration and error that often complicates the operation of mobile computer systems. In the short term, it seems unlikely that the technical limitations will be resolved. The following pages, therefore, argue that interface designers must consider means of reducing the impact of geographical location upon the operation of mobile computing devices.

Keywords: mobile computing, interface design, satellite communications, cellular networks.

1. Introduction

The Conference of European Telecommunication Authorities is currently working to 'harmonise' European networks for mobile communications. Similar initiatives have led to a digital mobile communications standard throughout North America. In Japan, there are plans for at least two different digital radio communications networks for mobile computing devices (Simon, 1996). These initiatives have encouraged hardware and software developers to invest in a vast array of hand-held and lap-top devices. Recent developments in the communications infrastructure, enables the users of these systems to access local and remote resources without being forced to connect to a physical telephone line. These systems are, in turn, posing new challenges for human computer interaction (Dix, 1996).

1.1. The Impact of Geography on Mobile Interaction

In order to understand the nature of the problems that frustrate interaction with mobile computer systems, it is important to have some idea of the underlying technology. Satellites offer a number of benefits for mobile interaction. Unlike radio systems, they do not suffer from the problems of multipath transmission. This occurs when signals 'bounce' off objects in the environment. This is a significant problem for interactive systems because mobile devices must then filter out any additional signals to recover the users' information. Unfortunately, satellites must filter and correct for atmospheric interference and for noise in space. There are further limitations. Geostationary satellites must maintain an orbit of approximately 36,000km in order to hold their position relative to the earth's surface. This incurs a half-second delay on transmissions which, in turn, affects the usability of mobile devices. For example, if an item of information is lost between the transmitter and the receiver then several seconds may go by before the missing item can be detected and corrected. Low-earth orbiting satellites avoid this delay but the user's device must then track the satellite's movement across the sky. Both forms of satellite communication currently suffer from a relatively low-bandwidth (8-20 Kbps). This limits the range of tasks that users can perform over these links.

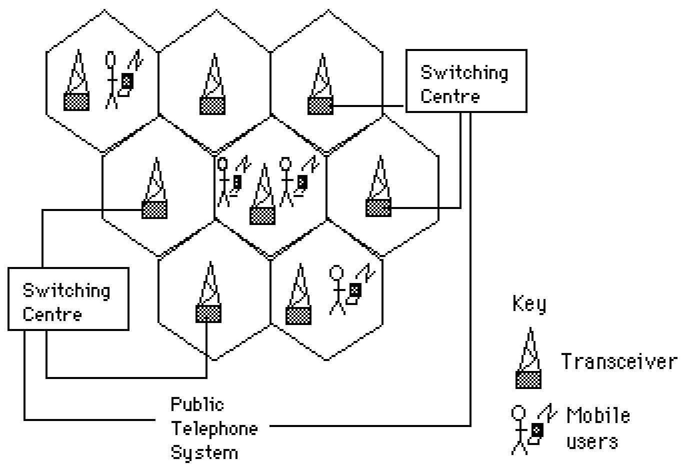
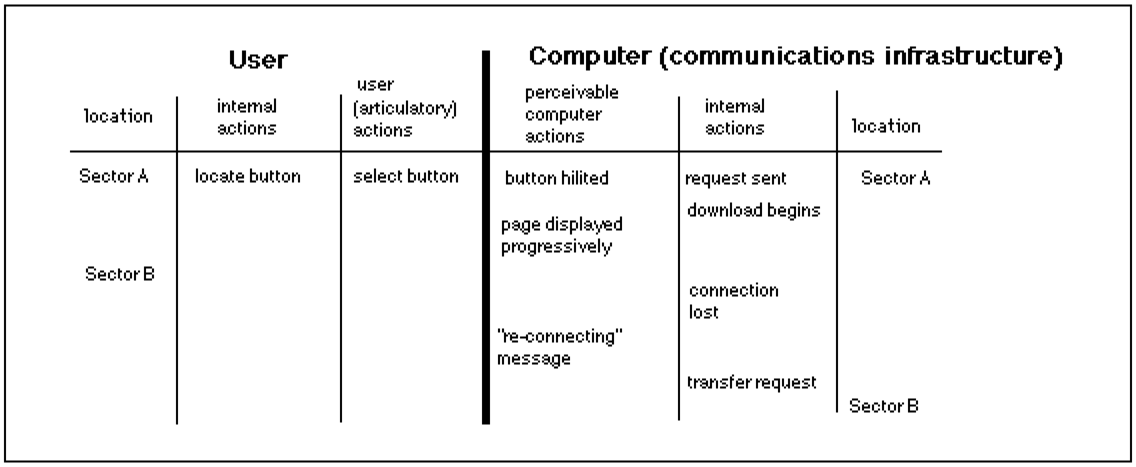


Figure 1: Cellular Communications Architecture

Figure 1 shows how cellular radio communications offer an alternative to satellite systems. Each cell has its own transceiver. As users move from one cell to another, their 'calls' are passed between transceivers. The idealised architecture shown in Figure 1 masks many of the problems that frustrate the development of mobile computer systems. There is a trade-off between the volume of information that a radio signal can carry and the distance that the signal will travel. High frequency signals carry more information but are susceptible to interference and dispersion. Low frequency signals carry less information but will travel over longer distances. Radio-based communication also suffers from: signal fade due to adverse atmospheric conditions; unintentional electromagnetic interference; interference from other devices using the same channel and variable signal strength due to movement of the device. Until such problems are addressed, users will continue to suffer the delays, broken connections and interruptions that frustrate mobile, human-computer interaction.

1.2. Notations for Interface Design

The technological barriers to digital communication are gradually being eroded. For instance, the Iridium project plans to use low-earth orbiting satellites to provide mobile communications from any point on the earth's surface. Until such systems have been fully developed, interface designers must continue to work within the constraints imposed by existing technology. The complexity of these systems makes it important that designers have some means of reasoning about mobile interaction. For example, existing task analysis techniques can be extended to represent the transfer of information between mobile systems (Johnson, Diaper and Long, 1985). Figure 2 achieves this by introducing location columns into Hix and Hartson's (1993) User Action Notation (UAN). The user is currently within one radio-cell. Their request for information is sent through the underlying infrastructure to a transceiver. Before the request can be completed, the user moves out of the cell. The connection is temporarily lost until the system re-establishes communication through another transceiver.



User			Computer (communications infrastructure)		
location	internal actions	user (articulatory) actions	perceivable computer actions	internal actions	location
Sector A	locate button	select button	button hilited	request sent	Sector A
				download begins	
			page displayed progressively		
Sector B					
				connection lost	
			"re-connecting" message		
				transfer request	
					Sector B

Figure 2: Extending UAN with Location Information

Unfortunately, a number of problems limit the utility of this notation. It provides no means of reasoning about temporal properties. This is important because the hand-over delay in Figure 2 might have a minimal effect if it lasted a few seconds. If it took several minutes then the "re-connecting" message might have to be reworded to provide more information about the cause of the delay. Temporal information can be represented using the extended XUAN notation (Gray and Johnson, 1995). There are, however, further problems. The column labelled 'Computer (communications infrastructure)' represents a considerable simplification. This does not capture the distinction between the state of the user's local machine and the communications network that provides access to a remote site. This is significant because the 'reconnecting message' can only be generated if the local machine detects the cause of the problem and takes appropriate action to re-establish the signal. Additional columns might be introduced into the table to represent the behaviour of local and remote machines but this would further reduce the tractability of UAN's tabular form.

A range of mathematically based, notations have been developed to reason about the problems of communication over mobile, distributed systems (Johnson 1996). Temporal extensions to formal specification techniques are a now a standard feature in the development of digital telecommunications systems (Austin and Parkin. 1993). Unfortunately, these notations have not been used to support interface design for mobile computing devices. This paper, therefore, demonstrates that a formal logic can be used to reason about the problems that frustrate human-computer interaction over mobile networks.

2. Using Logic To Represent Mobile Interaction

First order logic offers a number of advantages for the design of mobile human-computer interfaces. For instance, the use of a mathematical notation provides a link between interface design and the use of formal methods in the systems engineering of distributed systems. Logic formalisms also support prototyping through the use of concurrent execution environments, such as Prelog (Johnson, 1995). The following clauses illustrate this approach through the application of a Horn-clause notation to represent interaction with a mobile device. A user's request is successful if it is handled by a local machine. Alternatively, it is successful if the request is dispatched to an appropriate transceiver. In clause (2) the marshall proposition is used to indicate the process by which the users' request is packaged into a form that the transceiver can process. This proposition might be refined to introduce additional detail if the user's task required particular error correction or security features. Similarly, the term transceiver might be refined to represent a satellite transmission system.

initiate_transaction(user, request, local_machine) <=
input(user, request), handled(local_machine, request). (1)

initiate_transaction(user, request, transceiver) <=
input(user, request), not handled(local_machine, request),
covered(local_machine, location, transceiver),
marshall(request, message), dispatch(transceiver, message). (2)

The first clause states that a user's request is initiated if they input a request and it can be handled locally. The second clause states that a user's request is initiated if they input a request and it cannot be handled locally and a transceiver cover's the users location and the local machine dispatches the request through that transceiver.

The previous clauses suffer from a number of limitations. In particular, they focus on high level architectural features of the communications protocols that support mobile interaction. They do not illustrate the ways in which logic might also be used to analyse interface requirements during human- computer interaction with remote resources.

2.1 Supporting Display Design

The users of remote resources frequently become frustrated and may even abandon their requests if they cannot predict how long it will take to retrieve information (Johnson, 1995a). Such problems are particularly severe for the users of mobile systems which are dependent upon the underlying radio and satellite communications networks. The following clauses, therefore, builds upon (2) to specify that user confirmation is required if input cannot be handled locally. Even if a user cannot predict the amount of time that will be necessary for a particular request, they can at least determine whether or not a particular request is directed towards a remote machine.

```
confirmation_dialogue(user, request, local_machine) <=
input(user, request), not handled(local_machine, request),
display(local_machine, transfer_dialogue_box), input(user, confirm),
covered(local_machine, location, transceiver), marshall(request, message) ,
dispatch(transceiver, message).(3)
```

This states that a user's request is initiated if they input a request and it cannot be handled locally and the user issues input to confirm that they would like the request handled remotely and a transceiver covers the user's location and the local machine dispatches the request through that transceiver.

It is important to note that clause (3) does not specify the exact textual and graphical primitives that may be used to make up the transfer_dialogue_box. The following clauses illustrate the way in which such presentation information can be gradually introduced into high level specifications. Abstract images, such as the transfer_dialogue_box, can be described in terms of their component images. These component images can, in turn, be described in terms of the primitive graphical objects that are finally presented to the user (Johnson, 1995).

```
part(transfer_dialogue_box, left_arrow), (4)

line(left_arrow, 0.1,0.1,0.3,0.3). (5)

dimension(left_arrow, 0.3,0.3). (6)
```

The first clause states that the left arrow is part of the initiate transfer icon. The previous clauses also state that there is a line component of the left arrow and that the component has dimensions of (0.3, 0.3) relative to the entire display.

Previous clauses have shown that first order logic can be used to represent high-level interaction architectures for mobile systems. They have also illustrated the way in which this notation can be used to represent particular interface requirements. They have not, however, shown that logic can be used to analyse the systems behaviours which have a profound impact upon the usability of mobile computer systems.

2.2 Integrating Systems Engineering and User Requirements

A key requirement for the systems engineering of mobile computer systems is that the transceiver or satellite link should be able to unmarshall messages from remote users. This involves the use of error correction and encryption protocols. The transceiver must also determine whether the client and the destination are registered users of the service.

routing_service_request(transceiver, message) <=
unmarshall(message, local_machine, destination, request),
registered(transceiver, local_machine), registered(transceiver, destination),
marshall(request, message2), dispatch(destination, message2). (7)

This states that a request is serviced by a transceiver if it can unmarshall, or unpack, a message to identify both the target server and the local machine and both the server and local machines are registered with the transceiver and the message is forwarded to the target server.

Such systems requirements have a profound impact upon the users of mobile computing systems. For example, the previous clause does not specify what should happen if the transceiver or satellite link could not immediately dispatch the user's request to its intended destination. This would happen if, for instance, the destination were another mobile user who was currently out of range of a transceiver. Under such circumstances, the user who initiated the request would be left waiting for a response. Systems engineers and interface designers can exploit a number of techniques to mitigate such problems. For instance, many distributed systems now make use of binding services. In this architecture, the user does not specify a particular destination for their request. Instead, their input is passed to any available machine that is registered and authorised to handle it. If a particular destination is not available, for example if it is out of range, then the binder may forward the user's request to another machine. Such a service might be implemented through the higher level protocols of the switching centres, shown on Figure 1. This approach is typical of high-integrity systems where users require a high degree of assurance that their input will eventually be handled by a server.

service_request(binder, message)<=
unmarshall(message, local_machine, request),

offers(server, request), marshall(request, message2),
dispatch(server, message2).(8)

This states that a request is serviced by a transceiver if it can unmarshall, or unpack, a message to identify the local machine and the request and a server has registered an offer to fulfil such requests with the transceiver and the request is forwarded to the server.

This clause again illustrates the need to closely integrate interface design with the systems engineering of mobile systems. Many tasks can only be performed on particular machines. For example, if a person sent a message to another mobile user then that message must be directed towards a machine where that user has an account. Under such circumstances, it is important that interface designers provide users with mechanisms for directing binding services towards particular destinations.

3. What Happens When Things Go Wrong

This section builds upon the previous analysis and focuses more closely upon the impact that systems failures have upon human-computer interaction with mobile devices.

3.1 Transceiver Failure

Interaction with remote resources will be jeopardised if a user's local machine cannot access the gateway into the communications network. Tracking problems can prevent a mobile computer from locating low-orbit satellites. Atmospheric interference can disrupt communication with geostationary satellites. Adverse meteorological conditions and natural barriers can restrict access to the transceivers in cellular networks.

fail_transaction(user, request, transceiver) <=
input(user, request),
not handled(local_machine, request),
not covered(local_machine, location, transceiver). (9)

This states that a transaction fails if a user inputs a request which cannot be handled locally and the user's location is not covered by a transceiver.

Such failures create a number of problems. For example, there may be no means for the user to know whether or not they are covered by a transceiver at any particular moment during interaction. One solution would be for the user to issue periodic requests to find out whether the mobile device was within range of the network. The success or failure of a request might then be reported using the display design techniques described in clauses (4,5,6). This approach is hardly transparent; the user must continually monitor the state of their connection in order to determine whether a request will be successful. Alternatively, store and forward techniques can be used. This approach relies upon the local machine storing the user's request until the remote network can be contacted again. As soon as a connection is re-established the user's message can be processed. Under this systems, users cannot assume that their requests will be immediately dispatched through the communications system. It is only possible to assume that their input will eventually be successful. Unfortunately, such delays cannot easily be captured using first order logic. The following clause, therefore, introduces the × (read as 'eventually') operator from interval temporal logic. The syntax and semantics of this notation are described in Johnson (1995). In contrast, the remainder of this paper focuses on the application of the approach to support the development of mobile, human-computer interfaces.

store_and_forward_transaction(user, request, transceiver) <=
fail_transaction(user, request, transceiver),
not handled(local_machine, request),
<> covered(local_machine, location, transceiver),
marshall(request, message), dispatch(transceiver, message). (10)

A user's request is stored and later forwarded to a transceiver if a requested transaction fails and eventually, the user's location is covered by a transceiver and the message is forwarded to that site.

Store and forward techniques offers substantial benefits because they shield the user from the underlying systems architecture. Unfortunately, these benefits carry a cost. Users may not recognise the reasons for the delays that occur when their requests are stored for later transmission. It is, therefore, possible for the user to issue multiple requests in the belief that their original input has been lost. Each of the requests would then be processed as contact is resumed with the communications system. This problem might be avoided if users were warned that a transfer was stalled until the message can be dispatched.

store_and_forward_display(user, request, transceiver) <=
fail_transaction(user, request, transceiver),
not handled(local_machine, request)
display(local_machine, transfer_stalled) U
(covered(local_machine, location, transceiver),
marshall(request, message),
dispatch(transceiver, message)).(11)

This states that a user's request is stored and that a warning is displayed if a requested transaction fails and an icon is displayed to show that the request is stalled until the user's location is covered by a transceiver and the message is forwarded to that site.

The previous clause used the U (read as 'until') operator. This describes the duration of the delay that might occur before the mobile machine moves within range of the nearest transceiver. This illustrates an important benefit of the logic notation; designers are not forced to specify the exact, real- time duration of the warning. Such details can gradually be introduced as empirical evidence indicates the probable time required for a user to note the warning in a given application (Kuhmann, 1989).

3.2 Server Failure

Mobile human-computer interaction will also break-down if the communications infrastructure is in tact but the target resource is unavailable or unknown. The binding architecture, described in clause (8), is resilient to this form of failure because several machines may offer the same service. However, in a standard architecture the user may specify a particular destination server that cannot be recognised by the transceiver and its associated messaging system.

fail_server_request(transceiver, message) <=
unmarshall(message, server, local_machine, request),
not registered(server, local_machine). (12)

This states that a server fails to fulfil a request if a request is unmarshalled by a transceiver but the local machine is not registered with the anticipated server.

This clause again illustrates the need to integrate human-factors and systems engineering during the development of mobile computer systems. For instance, systems engineers must implement the underlying protocols to determine whether a server is not responding through system failure or through unexpected delays in the communications system. Human factors engineers must design displays to alert the user that their request has failed. The exact content and format of such warnings must be heavily influenced by the underlying communications protocols. In a transparent architecture with asynchronous requests or a store and forward mechanism, users may be very surprised to learn that a request has failed several minutes after it was issued.

fail_server_request_2(transceiver, message) <=
unmarshall(message, server, local_machine, request),
not registered(server, local_machine),
O(marshall(registration_failure, message),
dispatch(local_machine, message)). (13)

display_server_failure(user, local_machine) <=
unmarshall(message, local_machine, server, registration_failure),
display(local_machine, registration_failure) U
input(user, confirm),(14)

The first clause states that a server fails to fulfil a request if a request is unmarshalled by a transceiver but the local machine is not registered with the anticipated server and in the next interval a message is returned to the local machine to warn them that they are not registered for the server that they requested. The second clause states that a server failure is displayed to a user on a local machine if a message is unmarshalled on the local machine to provide notification of a registration failure and this warning is presented until the user confirms it.

Clause (13) exploits the O (read as 'next') temporal logic operator to specify that the user's local machine is alerted to the server failure as soon as possible after the failure is detected. This does not mean that the user's display will be instantly updated. If the transceiver or satellite loses contact with the local machine then the user may continue to interact with the local system as if the original command had been successful. Under a store and forward architecture, see clause (10), the local machine would then send these requests to the transceiver at the same time as the transceiver returned a warning about the server failure.

3.3 Binder Failure

As mentioned, binding architectures avoid the problems of server failure because more than one site may be able to service a user's request. Delays can be reduced if the underlying communications protocols route the user's input to the closest server or to a server with spare capacity. However, this does not avoid the problems that arise when no server can satisfy the user's request.

```
binding_service_failure(transceiver, message)<=
unmarshall(message, local_machine, request),
not offers(server, request). (15)
```

This states that a binding service fails if a local machine unmarshalls a request and no server has offered to satisfy that request.

The simplest solution to this problem would be to return a message that the input could not be handled. This is the approach described in (14). Alternatively, the local machine or the communications infrastructure could store the user's request in the hope that a server might eventually offer the service. This would cope with periodic systems failures where duplicate servers are not provided. The user could assume that their input would eventually be handled once the relevant resource became available. This approach characterises the mobile control systems that are used to schedule repair teams, for instance in the domestic gas industry. Supervisors enter the day's duties into their local system. This is, typically, a PC with a modem connection to mobile receivers in each of the teams' vehicles. New jobs are allocated to a crew whenever they complete a previous request.

```
binding_service_delay(binder, message) <=
unmarshall(message, local_machine, request),
not offers(server, request),
<> offers(server, request),
forward(server, request). (16)
```

This states that there is a delay in a binding service if a user's request is unmarshalled but in the present interval, no server offers to satisfy the request and eventually a server does become available and the request is forwarded.

Unfortunately, there are a number of problems with this approach. For instance, processing delays can lead to a backlog of requests. This problem can be avoided by specifying a time-out after which the user will be alerted that their request has blocked.

An alternative approach relies on the fact that no single binding service will know of all of the possible destinations for a user's request. In some cases, input may be passed to other cells or areas in order to find a suitable server. This is analogous to additional repair vehicles being called in from outside of a controller's area.

```
binding_service_forward_request(binder, message) <=
unmarshall(message, local_machine, request),
not offers(server, request),
covered(binder, location, binder2),
marshall(request, message),
dispatch(binder2, message). (17)
```

This states that a binder forwards a request if a message is unmarshalled and no server is registered to satisfy the enclosed request and the binder is covered by another binder and the first site marshalls the request and passed it on to the second site.

This approach has many benefits for the users of mobile computer systems. In particular, it can be used to perform load balancing in distributed applications. Tasks may migrate throughout the communications 'network'. However, a request may pass through many intermediate sites before it finds a host that is willing to perform any associated computation. This is similar to the way in which agents may migrate through conventional networks. Such approaches depend upon users finding sites that are willing and able to satisfy their tasks. Further work intends to fully explore the parallel between agent based systems and the emerging architectures for mobile human-computer interaction. The delays that arise through communications problems in mobile systems can be thought of as the time taken for agents to access remote resources.

4. Conclusion

This paper has argued that mobile devices pose new challenges for human-computer interaction. The technological limitations of radio and satellite systems can delay to user requests. These disruptions frequently lead to frustration and error (Walters, 1995). A number of international initiatives are currently devising solutions to these problems. For instance, by building a global network of low- orbit satellites. In the short-term, however, it is important that interface designers can mitigate the problems of mobile interaction. We have, therefore, shown that temporal extensions to first order logic can be used to analyse a number of different interaction architectures for mobile systems. Store and forward approaches have been compared to binding services. Transparent approaches that hide the underlying communications infrastructure from the user have been contrasted with approaches in which the user specifically directs their requests to particular sites. A critical theme in all of this has been the need to integrate interface design and systems engineering. Unless systems engineers appreciate the consequences of transmission delays then it may not be possible to implement the protocols and architectures that mitigate the usability problems of mobile interaction. Unless interface designers understand the underlying properties of modern communications then there is a danger that users will continue to face unexplained delays and periodic system failures.

Acknowledgements

Thanks are due to the members of the Glasgow Accident Analysis Group and the Glasgow Interactive Systems Group (GIST) who provided valuable support and encouragement for the work reported in this paper. This research has been supported by the UK Engineering and Physical Sciences Research Council, grants GR/L27800 and GR/K55042.

Appendix A: Table of Propositions

The following table provides informal descriptions of the propositions that are introduced in the paper and that are not given an informal description as part of the running text.

input(user, request) This is true if a user inputs a service request.

display(machine, display_element) This is true if a machine displays a graphical image denoted by display_element.

part(display_elem1, display_elem2) True if display_element2 is part of display_element1.

line(display_element, X, Y, X1, Y1). This is true if there is a line from Cartesian Co-ordinates (X, Y) to (X1, Y1) in a graphical image denoted by display_element.

dimension(display_element, X, Y). This is true if display_element occupies the X and Y dimensions of the screen where X and Y are in the range from 0.0 to 1.0.

handled(machine, request). True if a machine can handle or service a request.

covered(machine, location, transceiver) This is true if a machine falls within a transceiver's location.

marshall(request, message) This is true if a request is translated into a common message format that can be understood by the potential recipient.

unmarshall(message, local_machine, destination_machine, request) This is true if a message from a local_machine is translated into a request for a service on a destination_machine.

dispatch(machine, message) This is true if a message is dispatched to a destination machine.

registered(transceiver, machine) This is true if a machine is registered to access or be accessed by a transceiver.

offers(server, request) This is true if a server offers to service a particular request.

O(w) This is true iff a proposition w is true in the next state,

<>(w) This is true iff a proposition w is eventually true in some

future state. [w_1 U w_2] This is true iff a proposition w_1 is true until proposition w_2 is true.

References

S. Austin and G.I. Parkin, Formal Methods: A Survey, Division Of Information Technology And Computing, The National Physical Laboratory, Sponsored by the United Kingdom Department of Trade and Industry, 1993,

A. Dix, editor,CSCW Issues for Mobile and Teleworkers. Springer Verlag, Berlin, 1997.

P. Gray and C.W. Johnson, Requirements for Interface Design Notations. In P. Palanque and R. Bastide (eds.), The Design, Specification and Verification of Interactive Systems, 113-133. Springer Verlag, Berlin, 1995.

D. Hix and H.R. Hartson, Developing User Interfaces, John Wiley and Sons, London, 1993.

C.W. Johnson. Integrating Human Factors And Systems Engineering To Reduce The Risk Of Operator 'Error', Safety Science, 22(1-3):195-214. 1995.

C.W. Johnson, Time And The Web: Representing Temporal Properties Of Interaction With Distributed Systems. In M.A.R. Kirby and A.J. Dix and J.E. Finlay (eds.) People And Computers X: Proceedings of HCI'95, Cambridge University Press, Cambridge, United Kingdom, 39-50. 1995a.

C.W. Johnson The Impact of Working Environments on Human-Machine Interaction, Ergonomics, (39)3:512-530, 1996

P. Johnson, D. Diaper and J. Long, Task Analysis in Interactive Systems Design and Evaluation. In G. Johansen, G. Mancini and L. Martensson (eds.) Analysis, Design and Evaluation in Man-Machine Systems. Oxford University Press, 1985.

W. Kuhmann, Stress Inducing Properties Of System Response Times, Ergonomics, (32)3:271-280, 1989.

E. Simon, Distributed Information Systems. McGraw Hill, London, 1996.

R. Walters,Computer-Mediated Communications. Artech House, Boston, 1995.